

2024년 12월

-> STX, POP 시 POP을 두 개 함

-> 의도치 않은 분기 작동

```

58      POP     0, 1
59      INT     0, 12
60      LDA     0, 12
61      LOD     1, 12
62      POP     0, 5
63      ADDR    0, printf
64      CAL     0, 0
65      RET     0, 0
755_0| 66      .literal    12  printf|
81755| 67

```

Debug    interp

gen.c    **ASM a.asm**    lib.c    CM

```

1      INT    0, 20
2      SUP    0, main
3      RET    0, 0
4      multiply:
5      INT    0, 24
6      LDA    1, 20

```

```

74
75 {digit}+ { yyval = atoi(yytext); return(INTEGER_CONSTANT);
76 {digit}+\. {digit}+ { yyval = makeString(yytext); return(FLOAT_CONSTANT);
77 {letter}({letter}|{digit})* { return(checkIdentifier(yytext)); }
78 \"([^\n]|\\\"\\n)*\" { makeString(yytext); return(STRING_LITERAL); }
79 \'([^\n]|\\\'\\n)*\' { yyval = *(yytext + 1); return(CHARACTER_CONSTANT);
80

```

신텍스, 시멧릭스 트리는 생략하고, 어셈블리어 코드와 실행 결과를 분석하는 것에 집중하겠다.

테스트 파일은 testdir 폴더에, 컴파일러가 생성한 a.asm 파일은 testasm 폴더에 있다.

```

1  int multiply(int a, int b) {
2      int result;
3      result = 0;
4
5      while(a){
6          if(a%2)
7              result = result + b;
8          a = a/2;
9          b = b*2;
10     }
11
12     return result;
13 }
14
15 void main(){
16     int i;
17     i = 0;
18
19     i = multiply(120, 3);
20
21     printf("result=%d\n", i);
22 }
23
24

```

```

1  INT 0, 28
2  SUP 0, main
3  RET 0, 0
4
5  multiply:
6      INT 0, 24
7      LDA 1, 20
8      LITI 0, 0
9      STX 0, 1
10     POP 0, 1
11
12  L2:
13     LOD 1, 12
14     JPC 0, L3
15     LOD 1, 12
16     LITI 0, 2
17     MOD 0, 0
18     JPC 0, L4
19     LDA 1, 20
20     LOD 1, 20
21     LOD 1, 16
22     ADDI 0, 0
23     STX 0, 1
24     POP 0, 1
25
26  L4:
27     LDA 1, 12
28     LOD 1, 12
29     LITI 0, 2
30     DIVI 0, 0
31     STX 0, 1
32     POP 0, 1
33     LDA 1, 16
34     LOD 1, 16
35     LITI 0, 2
36     MULI 0, 0
37     STX 0, 1
38     POP 0, 1
39
40  L1:
41     JMP 0, L2
42
43  L3:
44     LDA 1, -4
45     LOD 1, 20
46     STO 0, 1
47     RET 0, 0
48     RET 0, 0
49
50  main:
51     INT 0, 16
52     LDA 1, 12
53     LITI 0, 0
54     STX 0, 1
55     POP 0, 1
56     LDA 1, 12
57     INT 0, 16
58     LITI 0, 120
59     LITI 0, 3
60     POP 0, 5
61     ADDR 0, multiply
62     CAL 0, 0
63     STX 0, 1
64     POP 0, 1
65     INT 0, 12
66     LDA 0, 12
67     LOD 1, 12
68     POP 0, 5
69     ADDR 0, printf
70     CAL 0, 0
71     RET 0, 0
72
73  .literal 12 "result=%d\n"

```

```

9: JPC 0, 33
10: LOD 1, 12
11: LITI 0, 2
12: MOD 0, 0
13: JPC 0, 20
14: LDA 1, 20
15: LOD 1, 20
16: LOD 1, 16
17: ADDI 0, 0
18: STX 0, 1
19: POP 0, 1
20: LDA 1, 12
21: LOD 1, 12
22: LITI 0, 2
23: DIVI 0, 0
24: STX 0, 1
25: POP 0, 1
26: LDA 1, 16
27: LOD 1, 16
28: LITI 0, 2
29: MULI 0, 0
30: STX 0, 1
31: POP 0, 1
32: JMP 0, 8
33: LDA 1, -4
34: LOD 1, 20
35: STO 0, 1
36: RET 0, 0
37: RET 0, 0
38: INT 0, 16
39: LDA 1, 12
40: LITI 0, 0
41: STX 0, 1
42: POP 0, 1
43: LDA 1, 12
44: INT 0, 16
45: LITI 0, 120
46: LITI 0, 3
47: POP 0, 5
48: ADDR 0, 3
49: CAL 0, 0
50: STX 0, 1
51: POP 0, 1
52: INT 0, 12
53: LDA 0, 12
54: LOD 1, 12
55: POP 0, 5
56: ADDR 0, -1
57: CAL 0, 0
58: RET 0, 0

```

```

start execution
result=360
end execution
kh@ThinkPad-T16g2: ~/comp

```

잘 완성되었음을 마지막으로 검증하는 목적을 겸하여 교재 10장 첫 번째 예제인 multiply 프로그램으로 테스트를 해 본 결과, 컴파일러가 교재와 똑같은 흐름의 어셈블리어 코드를 생성했고, 실행된 인터프리터가 의도한 대로 올바르게 출력함을 확인했다.

```

1  int a[10];
2
3  void sort(int l, int r) {
4      int i, j, x, w, k;
5      i = l;
6      j = r;
7
8      x = a[(l+r)/2];
9
10     do{
11         while(a[i] < x){
12             i++;
13         }
14         while(x < a[j]){
15             j--;
16         }
17
18         if(i <= j){
19             w = a[i];
20             a[i] = a[j];
21             a[j] = w;
22             i++;
23             j--;
24         }
25     } while (i <= j);
26
27     if (l < j){
28         sort(l, j);
29     }
30     if (i < r){
31         sort(i, r);
32     }
33 }
34
35 void main(){
36     int k;
37
38     a[0] = 0;
39     a[1] = 1;
40     a[2] = 3;
41     a[3] = 5;
42     a[4] = 7;
43     a[5] = 9;
44     a[6] = 2;
45     a[7] = 4;
46     a[8] = 6;
47     a[9] = 8;
48
49     for(k=0; k<10; k++){
50         printf("%d ", a[k]);
51     }
52     printf("\n");
53
54     sort(0, 9);
55
56     for(k=0; k<10; k++){
57         printf("%d ", a[k]);
58     }
59     printf("\n");
60
61 }
62

```

```

237: LITI 0, 4
238: MULI 0, 0
239: OFFSET 0, 0
240: LDI 0, 1
241: POP 0, 5
242: ADDR 0, printf
243: CAL 0, 0
244: L13: LOD 1, 12
245: LDA 1, 12
246: LDX 0, 1
247: INCI 0, 0
248: STO 0, 1
249: POP 0, 1
250: JMP 0, L14
251:
252: L15: INT 0, 12
253: LDA 0, 60
254: POP 0, 4
255: ADDR 0, printf
256: CAL 0, 0
257: INT 0, 12
258: LITI 0, 0
259: LITI 0, 9
260: POP 0, 5
261: ADDR 0, sort
262: CAL 0, 0
263: LDA 1, 12
264: LITI 0, 0
265: STX 0, 1
266: POP 0, 1
267: L17: LOD 1, 12
268: LITI 0, 10
269: LSSI 0, 0
270: JPC 0, L18
271: INT 0, 12
272: LDA 0, 68
273: LDA 0, 12
274: LOD 1, 12
275: LITI 0, 4
276: MULI 0, 0
277: OFFSET 0, 0
278: LDI 0, 1
279: POP 0, 5
280: ADDR 0, printf
281: CAL 0, 0
282: L16: LOD 1, 12
283: LDA 1, 12
284: LDX 0, 1
285: INCI 0, 0
286: STO 0, 1
287: POP 0, 1
288: JMP 0, L17
289: L18: INT 0, 12
290: LDA 0, 76
291: POP 0, 4
292: ADDR 0, printf
293: CAL 0, 0
294: RET 0, 0
295: .literal 52 "%d "
296: .literal 60 "\n"
297: .literal 68 "%d "
298: .literal 76 "\n"

```

```

229: LDA 1, 12
230: LDX 0, 1
231: INCI 0, 0
232: STO 0, 1
233: POP 0, 1
234: JMP 0, 213
235: INT 0, 12
236: LDA 0, 60
237: POP 0, 4
238: ADDR 0, -1
239: CAL 0, 0
240: INT 0, 12
241: LITI 0, 0
242: LITI 0, 9
243: POP 0, 5
244: ADDR 0, 3
245: CAL 0, 0
246: LDA 1, 12
247: LITI 0, 0
248: STX 0, 1
249: POP 0, 1
250: LOD 1, 12
251: LITI 0, 10
252: LSSI 0, 0
253: JPC 0, 272
254: INT 0, 12
255: LDA 0, 68
256: LDA 0, 12
257: LOD 1, 12
258: LITI 0, 4
259: MULI 0, 0
260: OFFSET 0, 0
261: LDI 0, 1
262: POP 0, 5
263: ADDR 0, -1
264: CAL 0, 0
265: LOD 1, 12
266: LDA 1, 12
267: LDX 0, 1
268: INCI 0, 0
269: STO 0, 1
270: POP 0, 1
271: JMP 0, 250
272: INT 0, 12
273: LDA 0, 76
274: POP 0, 4
275: ADDR 0, -1
276: CAL 0, 0
277: RET 0, 0

```

```

start execution
0 1 3 5 7 9 2 4 6 8
0 1 2 3 4 5 6 7 8 9
end execution

```

교재 10장의 두 번째 예제인 quicksort 프로그램으로 테스트를 해 본 결과, 컴파일러가 교재와 똑같은 흐름의 어셈블리어 코드를 생성했고, 실행된 인터프리터가 의도한 대로 올바르게 출력함을 확인했다.

The screenshot displays a development environment with three panels:

- Source Code (3\_helloworld.c):**

```

1 void helloworld(){
2     int i;
3
4     for(i = 0; i < 10; i++){
5         printf("%d : hello, world\n", i+1);
6     }
7 }
8
9 void main(){
10     helloworld();
11 }

```
- Assembly Code (a.asm):**

```

1      INT    0, 36
2      SUP    0, main
3      RET    0, 0
4 helloworld:
5      INT    0, 16
6      LDA    1, 12
7      LITI   0, 0
8      STX    0, 1
9      POP    0, 1
10
11 L2:
12      LOD    1, 12
13      LITI   0, 10
14      LSSI   0, 0
15      JPC    0, L3
16      INT    0, 12
17      LDA    0, 12
18      LOD    1, 12
19      LITI   0, 1
20      ADDI   0, 0
21      POP    0, 5
22      ADDR   0, printf
23      CAL    0, 0
24
25 L1:
26      LOD    1, 12
27      LDA    1, 12
28      LDX    0, 1
29      INCI    0, 0
30      STO    0, 1
31      POP    0, 1
32      JMP    0, L2
33
34 L3:
35      RET    0, 0
36
37 main:
38      INT    0, 12
39      INT    0, 12
40      POP    0, 3
41      ADDR   0, helloworld
42      CAL    0, 0
43      RET    0, 0
44
45 .literal   12  "%d : hello, world\n"

```
- Terminal:**

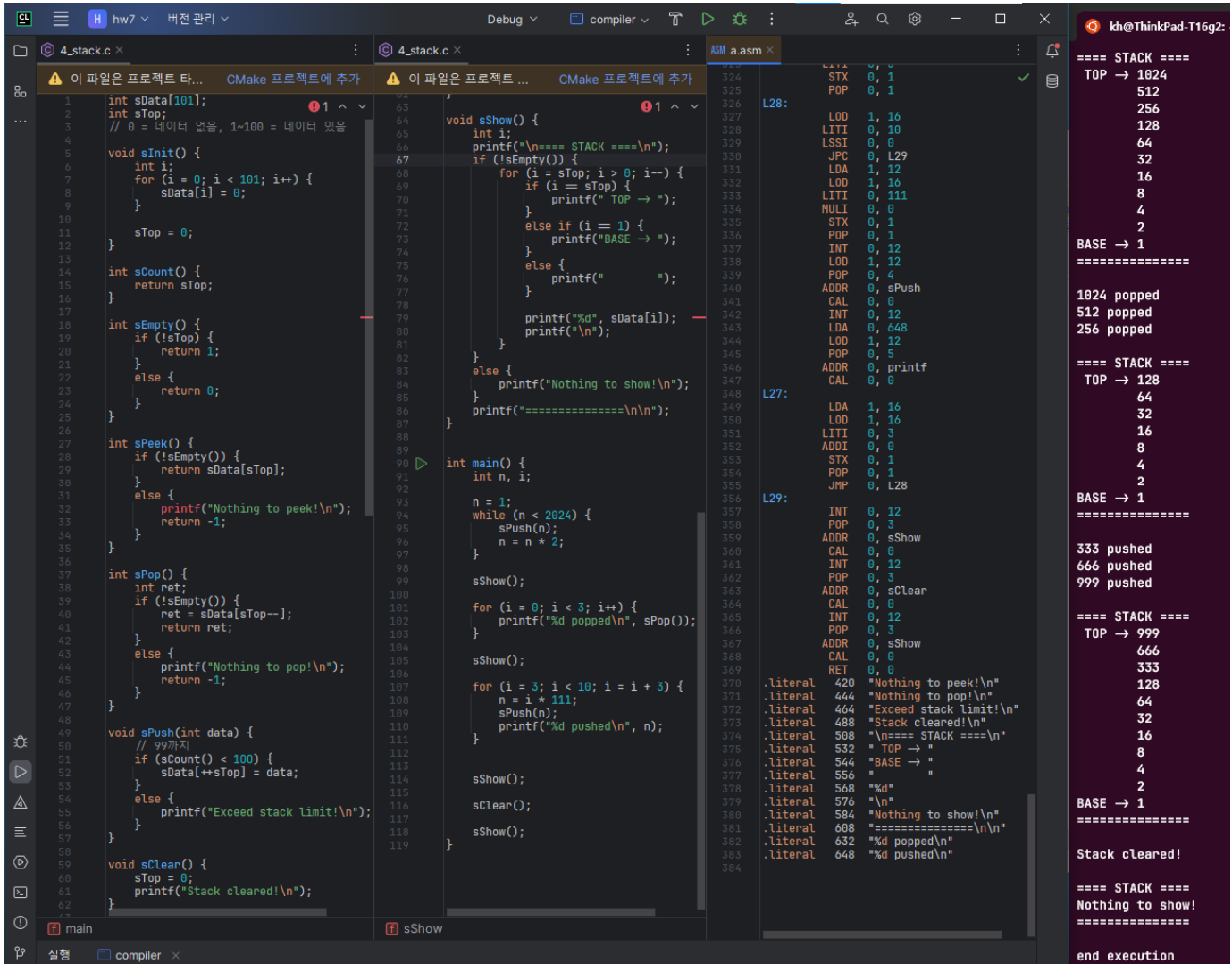
```

2: malloc -2
3: scanf -3
4: main 28
5: helloworld 3
6: L2 8
7: L3 27
8: L1 20
===== code =====
0: INT 0,36
1: SUP 0,28
2: RET 0,0
3: INT 0,16
4: LDA 1,12
5: LITI 0,0
6: STX 0,1
7: POP 0,1
8: LOD 1,12
9: LITI 0,10
10: LSSI 0,0
11: JPC 0,27
12: INT 0,12
13: LDA 0,12
14: LOD 1,12
15: LITI 0,1
16: ADDI 0,0
17: POP 0,5
18: ADDR 0,-1
19: CAL 0,0
20: LOD 1,12
21: LDA 1,12
22: LDX 0,1
23: INCI 0,0
24: STO 0,1
25: POP 0,1
26: JMP 0,8
27: RET 0,0
28: INT 0,12
29: INT 0,12
30: POP 0,3
31: ADDR 0,3
32: CAL 0,0
33: RET 0,0
start execution
1 : hello, world
2 : hello, world
3 : hello, world
4 : hello, world
5 : hello, world
6 : hello, world
7 : hello, world
8 : hello, world
9 : hello, world
10 : hello, world
end execution

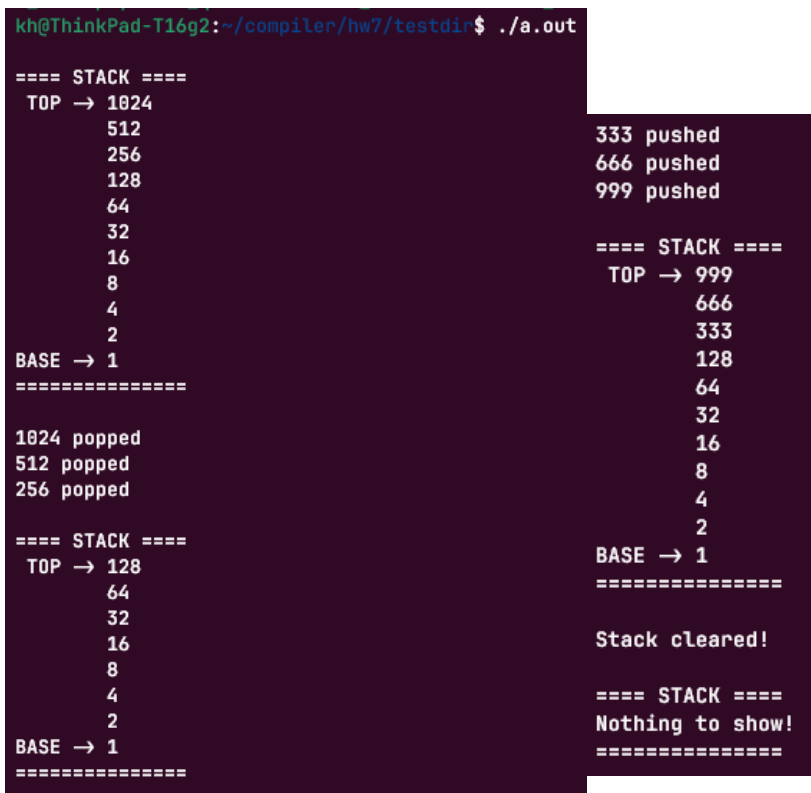
```

간단한 헬로월드 코드이다.

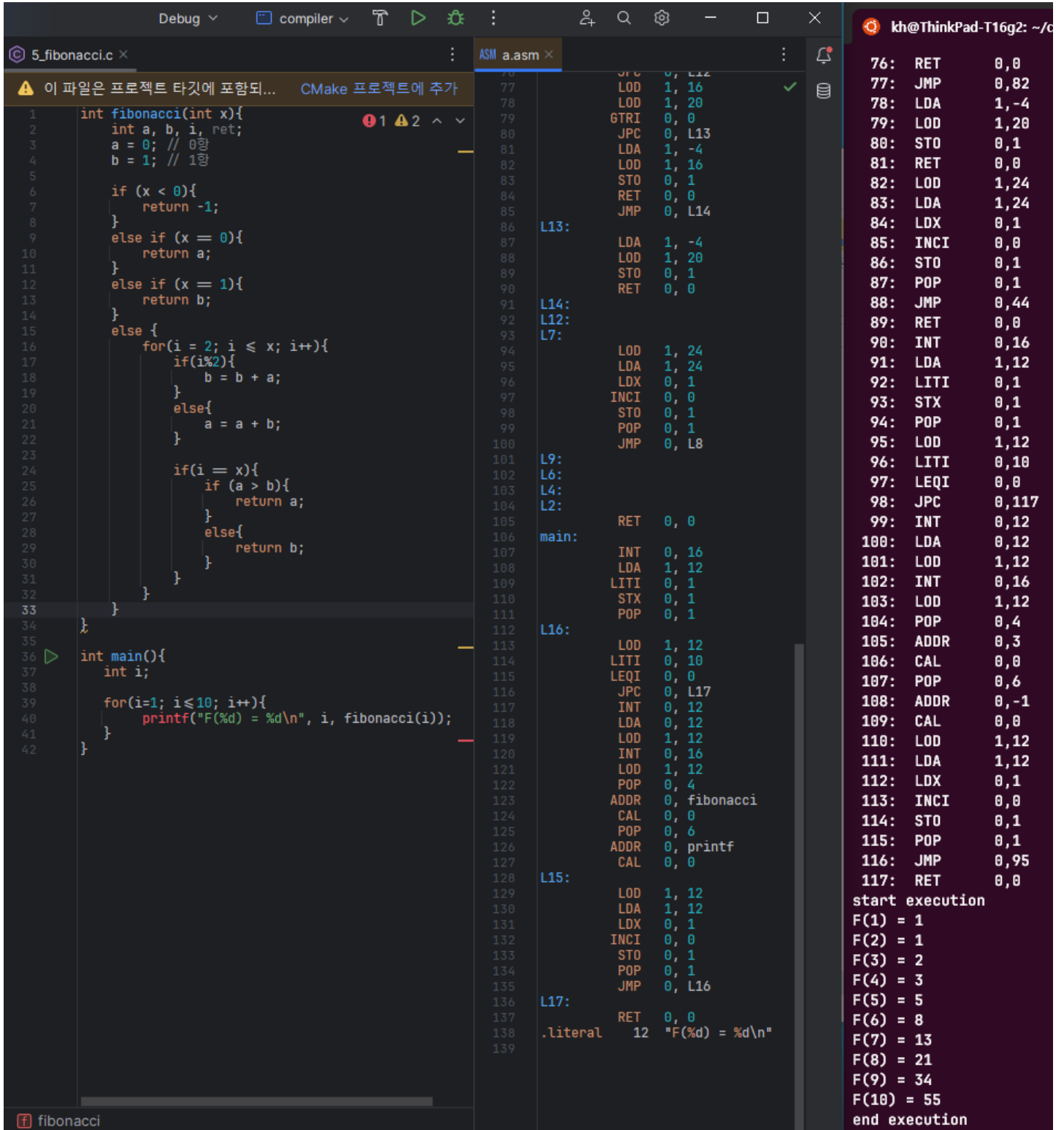
#### 4\_stack.c



만들어진 컴파일러가 지원하는 문법 안에서 작동하게끔 스택 자료구조를 구현했다.



gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.



```

1  int fibonacci(int x){
2      int a, b, i, ret;
3      a = 0; // 0항
4      b = 1; // 1항
5
6      if (x < 0){
7          return -1;
8      }
9      else if (x == 0){
10         return a;
11     }
12     else if (x == 1){
13         return b;
14     }
15     else {
16         for(i = 2; i ≤ x; i++){
17             if(i%2){
18                 b = b + a;
19             }
20             else{
21                 a = a + b;
22             }
23
24             if(i == x){
25                 if (a > b){
26                     return a;
27                 }
28                 else{
29                     return b;
30                 }
31             }
32         }
33     }
34 }
35
36 int main(){
37     int i;
38
39     for(i=1; i≤10; i++){
40         printf("F(%d) = %d\n", i, fibonacci(i));
41     }
42 }

```

```

76: RET 0,0
77: JMP 0,82
78: LDA 1,-4
79: LOD 1,20
80: STO 0,1
81: RET 0,0
82: LOD 1,24
83: LDA 1,24
84: LDX 0,1
85: INCI 0,0
86: STO 0,1
87: POP 0,1
88: JMP 0,44
89: RET 0,0
90: INT 0,16
91: LDA 1,12
92: LITI 0,1
93: STX 0,1
94: POP 0,1
95: LOD 1,12
96: LITI 0,10
97: LEQI 0,0
98: JPC 0,117
99: INT 0,12
100: LDA 0,12
101: LOD 1,12
102: INT 0,16
103: LOD 1,12
104: POP 0,4
105: ADDR 0,3
106: CAL 0,0
107: POP 0,6
108: ADDR 0,-1
109: CAL 0,0
110: LOD 1,12
111: LDA 1,12
112: LDX 0,1
113: INCI 0,0
114: STO 0,1
115: POP 0,1
116: JMP 0,95
117: RET 0,0

```

```

start execution
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55
end execution

```

피보나치 수열을 출력하는 프로그램이다.

```

kh@ThinkPad-T16g2:~/compiler/hw7/testdir$ ./a.out
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55

```

gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.

```

1  int gcd(int x, int y) {
2      int a, b, tmp;
3      a = x;
4      b = y;
5
6      while (b != 0) {
7          tmp = a % b;
8          a = b;
9          b = tmp;
10     }
11     return a;
12 }
13
14 int lcm(int x, int y) {
15     int g;
16     g = gcd(x, y);
17     return x*y/g;
18 }
19
20 void main() {
21     int a, b;
22     a = 15;
23     b = 50;
24
25     printf("gcd(%d, %d) = %d\n", a, b, gcd(a, b));
26     printf("lcm(%d, %d) = %d\n", a, b, lcm(a, b));
27 }

```

```

37     LOD     1, 20
38     STO     0, 1
39     RET     0, 0
40
41     lcm:
42         INT     0, 24
43         LDA     1, 20
44         INT     0, 16
45         LOD     1, 12
46         LOD     1, 16
47         POP     0, 5
48         ADDR     0, gcd
49         CAL     0, 0
50         STX     0, 1
51         POP     0, 1
52         LDA     1, -4
53         LOD     1, 12
54         LOD     1, 16
55         MULI     0, 0
56         LOD     1, 20
57         DIVI     0, 0
58         STO     0, 1
59         RET     0, 0
60
61     main:
62         INT     0, 20
63         LDA     1, 12
64         LITI     0, 15
65         STX     0, 1
66         POP     0, 1
67         LDA     1, 16
68         LITI     0, 50
69         STX     0, 1
70         POP     0, 1
71         INT     0, 12
72         LDA     0, 12
73         LOD     1, 12
74         LOD     1, 16
75         INT     0, 16
76         LOD     1, 12
77         LOD     1, 16
78         POP     0, 5
79         ADDR     0, gcd
80         CAL     0, 0
81         POP     0, 7
82         ADDR     0, printf
83         CAL     0, 0
84         INT     0, 12
85         LDA     0, 36
86         LOD     1, 12
87         LOD     1, 16
88         INT     0, 16
89         LOD     1, 12
90         LOD     1, 16
91         POP     0, 5
92         ADDR     0, lcm
93         CAL     0, 0
94         POP     0, 7
95         ADDR     0, printf
96         CAL     0, 0
97         RET     0, 0
98     .literal    12    "gcd(%d, %d) = %d\n"
99     .literal    36    "lcm(%d, %d) = %d\n"
100

```

```

41: POP     0, 5
42: ADDR     0, 3
43: CAL     0, 0
44: STX     0, 1
45: POP     0, 1
46: LDA     1, -4
47: LOD     1, 12
48: LOD     1, 16
49: MULI     0, 0
50: LOD     1, 20
51: DIVI     0, 0
52: STO     0, 1
53: RET     0, 0
54: RET     0, 0
55: INT     0, 20
56: LDA     1, 12
57: LITI     0, 15
58: STX     0, 1
59: POP     0, 1
60: LDA     1, 16
61: LITI     0, 50
62: STX     0, 1
63: POP     0, 1
64: INT     0, 12
65: LDA     0, 12
66: LOD     1, 12
67: LOD     1, 16
68: INT     0, 16
69: LOD     1, 12
70: LOD     1, 16
71: POP     0, 5
72: ADDR     0, 3
73: CAL     0, 0
74: POP     0, 7
75: ADDR     0, -1
76: CAL     0, 0
77: INT     0, 12
78: LDA     0, 36
79: LOD     1, 12
80: LOD     1, 16
81: INT     0, 16
82: LOD     1, 12
83: LOD     1, 16
84: POP     0, 5
85: ADDR     0, 36
86: CAL     0, 0
87: POP     0, 7
88: ADDR     0, -1
89: CAL     0, 0
90: RET     0, 0

```

```

start execution
gcd(15, 50) = 5
lcm(15, 50) = 150
end execution

```

최대공약수, 최소공배수를 구하는 프로그램이다.

```

kh@ThinkPad-T16g2:~/compiler/hw7/testdir$ ./a.out
gcd(15, 50) = 5
lcm(15, 50) = 150

```

gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.



## 7\_prime.c

The image shows a development environment with three main panes. The left pane displays the C source code for `7_prime.c`, which includes a `prime` function and a `main` function. The middle pane shows the corresponding assembly code, with labels like `L6:`, `L12:`, `L13:`, and `main:`. The right pane shows the execution output, starting with `start execution` and listing prime numbers: `2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89`, followed by `end execution` and the command prompt `kh@ThinkPad-T16g2:~/compiler/hw7/cmake-build-debug$`.

주어진 수 이하의 소스를 찾는 프로그램이다.

A terminal window showing the command `kh@ThinkPad-T16g2:~/compiler/hw7/testdir$ ./a.out` and its output: `2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89` on the first line and `2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113` on the second line.

gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.



The image shows a code editor with three panels. The left panel displays the C source code for a bubble sort program. The middle panel shows the corresponding assembly code generated by a compiler. The right panel shows the execution output of the program.

```

1 void bubbleSort(int arr[], int n) {
2     int i, j, temp;
3     for (i = 0; i < n - 1; i++) {
4         for (j = 0; j < n - i - 1; j++) {
5             if (arr[j] > arr[j + 1]) {
6                 temp = arr[j];
7                 arr[j] = arr[j + 1];
8                 arr[j + 1] = temp;
9             }
10        }
11    }
12 }

13 void printArray(int arr[], int size) {
14     int i;
15     for (i = 0; i < size; i++) {
16         printf("%d ", arr[i]);
17     }
18     printf("\n");
19 }

20 int main() {
21     int arr[7], n;
22
23     arr[0] = 64;
24     arr[1] = 34;
25     arr[2] = 90;
26     arr[3] = 12;
27     arr[4] = 22;
28     arr[5] = 11;
29     arr[6] = 25;
30
31     n = sizeof(arr)/sizeof(int);
32
33     printArray(arr, n);
34     bubbleSort(arr, n);
35     printArray(arr, n);
36
37     return 0;
38 }

```

The assembly code (ASM a.asm) shows the translation of the C code into machine instructions. It includes instructions for loading, storing, comparing, and swapping array elements, as well as printing the array.

The execution output shows the initial array [64, 34, 90, 12, 22, 11, 25] and the sorted array [11, 12, 22, 25, 34, 64, 90].

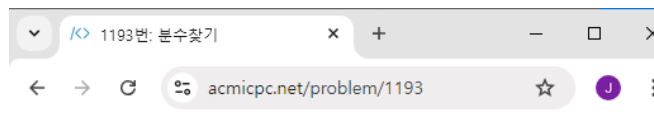
버블소트를 하는 프로그램이다.

```

kh@ThinkPad-T16g2:~/compiler/hw7/testdir$ ./a.out
64 34 90 12 22 11 25
11 12 22 25 34 64 90

```

gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.



## 분수찾기

성공



5 실버 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.5 초 (추가 시간 없음)	256 MB	135347	69121	59357	51.850%

## 문제

무한히 큰 배열에 다음과 같이 분수들이 적혀있다.

1/1	1/2	1/3	1/4	1/5	...
2/1	2/2	2/3	2/4	...	...
3/1	3/2	3/3	...	...	...
4/1	4/2	...	...	...	...
5/1	...	...	...	...	...
...	...	...	...	...	...

이와 같이 나열된 분수들을  $1/1 \rightarrow 1/2 \rightarrow 2/1 \rightarrow 3/1 \rightarrow 2/2 \rightarrow \dots$  과 같은 지그재그 순서로 차례대로 1번, 2번, 3번, 4번, 5번, ... 분수라고 하자.

$x$ 가 주어졌을 때,  $x$ 번째 분수를 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에  $x(1 \leq x \leq 10,000,000)$ 가 주어진다.

## 출력

첫째 줄에 분수를 출력한다.

### 예제 입력 1 복사

1

### 예제 입력 2 복사

2

### 예제 입력 3 복사

3

### 예제 입력 4 복사

4

### 예제 입력 5 복사

5

### 예제 입력 6 복사

6

### 예제 입력 7 복사

7

### 예제 입력 8 복사

8

### 예제 입력 9 복사

9

### 예제 출력 1 복사

1/1

### 예제 출력 2 복사

1/2

### 예제 출력 3 복사

2/1

### 예제 출력 4 복사

3/1

### 예제 출력 5 복사

2/2

### 예제 출력 6 복사

1/3

### 예제 출력 7 복사

1/4

### 예제 출력 8 복사

2/3

### 예제 출력 9 복사

3/2

<https://www.acmicpc.net/problem/1193> 에 대한 풀이이다.

Debug ▾ compiler ▾ ⋮

9\_boj1193.c × ASM a.asm ×

이 파일은 프... CMake 프로젝트에 추가

1 void fun(int n) {  
2 int i, rc, row, col;  
3  
4 char prev, end;  
5 int endcount;  
6  
7 rc = 2;  
8 row = 1;  
9 col = 1;  
10  
11 prev = 0;  
12 end = 1;  
13  
14 endcount = 2;  
15  
16 for (i = 1; i < n; i++) {  
17 if (endcount == rc) {  
18 if (rc % 2 == 0) {  
19 col++;  
20 }  
21 else {  
22 row++;  
23 }  
24  
25 rc++;  
26 endcount = 2;  
27 }  
28 else {  
29 if (rc % 2 == 0) {  
30 col++;  
31 row--;  
32 }  
33 else {  
34 row++;  
35 col--;  
36 }  
37 endcount++;  
38 }  
39 }  
40  
41 printf("%d/%d", row, col);  
42 }  
43  
44 int main() {  
45 int i;  
46  
47 for (i = 1; i < 10; i++) {  
48 printf("Answer %d : ", i);  
49 fun(i);  
50 printf("\n");  
51 }  
52 }

107 POP 0, 1  
108  
109 L9: LOD 1, 40  
110 LDA 1, 40  
111 LDX 0, 1  
112 INCI 0, 0  
113 STO 0, 1  
114 POP 0, 1  
115  
116 L7: L1:  
117 LOD 1, 16  
118 LDA 1, 16  
119 LDX 0, 1  
120 INCI 0, 0  
121 STO 0, 1  
122 POP 0, 1  
123 JMP 0, L2  
124  
125 L3: INT 0, 12  
126 LDA 0, 12  
127 LOD 1, 24  
128 LOD 1, 28  
129 POP 0, 6  
130 ADDR 0, printf  
131 CAL 0, 0  
132 RET 0, 0  
133  
134 main: INT 0, 16  
135 LDA 1, 12  
136 LITI 0, 1  
137 STX 0, 1  
138 POP 0, 1  
139  
140 L11: LOD 1, 12  
141 LITI 0, 10  
142 LSSI 0, 0  
143 JPC 0, L12  
144 INT 0, 12  
145 LDA 0, 20  
146 LOD 1, 12  
147 POP 0, 5  
148 ADDR 0, printf  
149 CAL 0, 0  
150 INT 0, 12  
151 LOD 1, 12  
152 POP 0, 4  
153 ADDR 0, fun  
154 CAL 0, 0  
155 INT 0, 12  
156 LDA 0, 36  
157 POP 0, 4  
158 ADDR 0, printf  
159 CAL 0, 0  
160  
161 L10: LOD 1, 12  
162 LDA 1, 12  
163 LDX 0, 1  
164 INCI 0, 0  
165 STO 0, 1  
166 POP 0, 1  
167 JMP 0, L11  
168  
169 L12: RET 0, 0  
170 .literal 12 "%d/%d"  
171 .literal 20 "Answer %d : "  
172 .literal 36 "\n"  
173

kh@ThinkPad-T16g2: ~/co

112: POP 0,1  
113: JMP 0,32  
114: INT 0,12  
115: LDA 0,12  
116: LOD 1,24  
117: LOD 1,28  
118: POP 0,6  
119: ADDR 0,-1  
120: CAL 0,0  
121: RET 0,0  
122: INT 0,16  
123: LDA 1,12  
124: LITI 0,1  
125: STX 0,1  
126: POP 0,1  
127: LOD 1,12  
128: LITI 0,10  
129: LSSI 0,0  
130: JPC 0,154  
131: INT 0,12  
132: LDA 0,20  
133: LOD 1,12  
134: POP 0,5  
135: ADDR 0,-1  
136: CAL 0,0  
137: INT 0,12  
138: LOD 1,12  
139: POP 0,4  
140: ADDR 0,3  
141: CAL 0,0  
142: INT 0,12  
143: LDA 0,36  
144: POP 0,4  
145: ADDR 0,-1  
146: CAL 0,0  
147: LOD 1,12  
148: LDA 1,12  
149: LDX 0,1  
150: INCI 0,0  
151: STO 0,1  
152: POP 0,1  
153: JMP 0,127  
154: RET 0,0  
start execution  
Answer 1 : 1/1  
Answer 2 : 1/2  
Answer 3 : 2/1  
Answer 4 : 3/1  
Answer 5 : 2/2  
Answer 6 : 1/3  
Answer 7 : 1/4  
Answer 8 : 2/3  
Answer 9 : 3/2  
end execution

```
kh@ThinkPad-T16g2:~/compiler/hw7/testdir$ ./a.out
Answer 1 : 1/1
Answer 2 : 1/2
Answer 3 : 2/1
Answer 4 : 3/1
Answer 5 : 2/2
Answer 6 : 1/3
Answer 7 : 1/4
Answer 8 : 2/3
Answer 9 : 3/2
```

gcc로 컴파일한 실행파일과 동일하게 작동하는 것을 확인했다.